**Instructions**

It is expected that you are at an internet connected computer and logged into the Canvas Conference that will be started at the beginning of the exam. Check for any recent Canvas announcement containing last minute information. You will be given further instructions live during the conference. This is an open-book, open-note exam. You may use a calculator or computer and any notes/lab notebook that you have kept during this class.

**You MAY NOT collaborate or communicate** with another person in any way. If another person contacts you for answers, you must ignore that request and report it to me (I will ask for evidence).

Exam problems will require you to edit the provided Word document file. Some problems may require you to work out a problem and show your work. You will have to use your own scratch paper for this. Paste the picture in the Word doc or upload the picture of your work with the exam.

At the end of the exam you must upload the edited doc and any pictures taken of your hand done work. Upload all files individually to the Canvas assignment. Do not zip them together. There will be a fifteen-minute window after the end of the exam to get everything uploaded to the Canvas assignment.

**For problems requiring you to show your work:**

* Using scratch paper, write the problem number and all work showing your solution to the problem
* Take a picture of the work and paste it into the Word doc or upload this picture with the Word doc
* The file name of the doc and/or pictures do not matter

**STATEMENT ON ETHICS, PROFESSIONALISM, AND CONDUCT OF ENGINEERING STUDENTS**

**COLLEGE OF ENGINEERING**

**THE UNIVERSITY OF TEXAS AT ARLINGTON**

The College cannot and will not tolerate any form of academic dishonesty by its students.  This includes, but is not limited to 1) cheating on examination, 2) plagiarism, or 3) collusion.

Definitions:

A.  Cheating on an examination includes:

1.   Copying from another's paper, any means of communication with another during examination, giving aid to or receiving aid from another during examination;

2.   Using any material during examination that is unauthorized by the proctor;

3.   Taking or attempting to take an examination for another student or allowing another student to take or attempt to take an examination for oneself.

4.   Using, obtaining, or attempting to obtain by any means the whole or any part of an un-administered examination.

B.   Plagiarism is the unacknowledged incorporation of another's work into work which the student offers for credit.

C.  Collusion is the unauthorized collaboration of another in preparing work that a student offers for credit.

I have read and I understand the above statement.

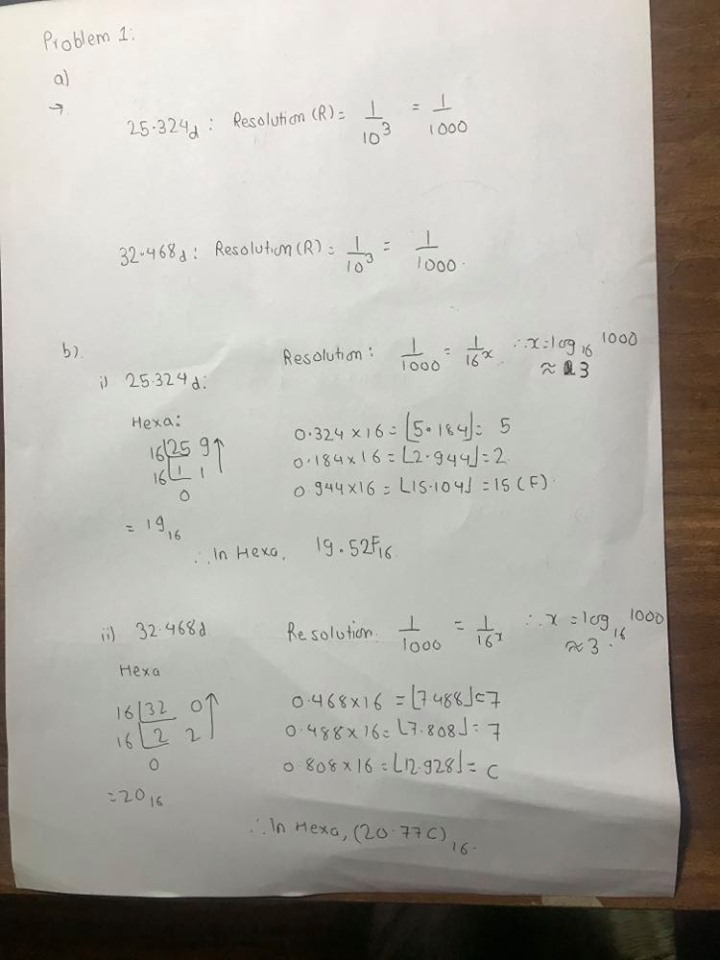
  Typed First and Last Name: **Prabesh Humagain**

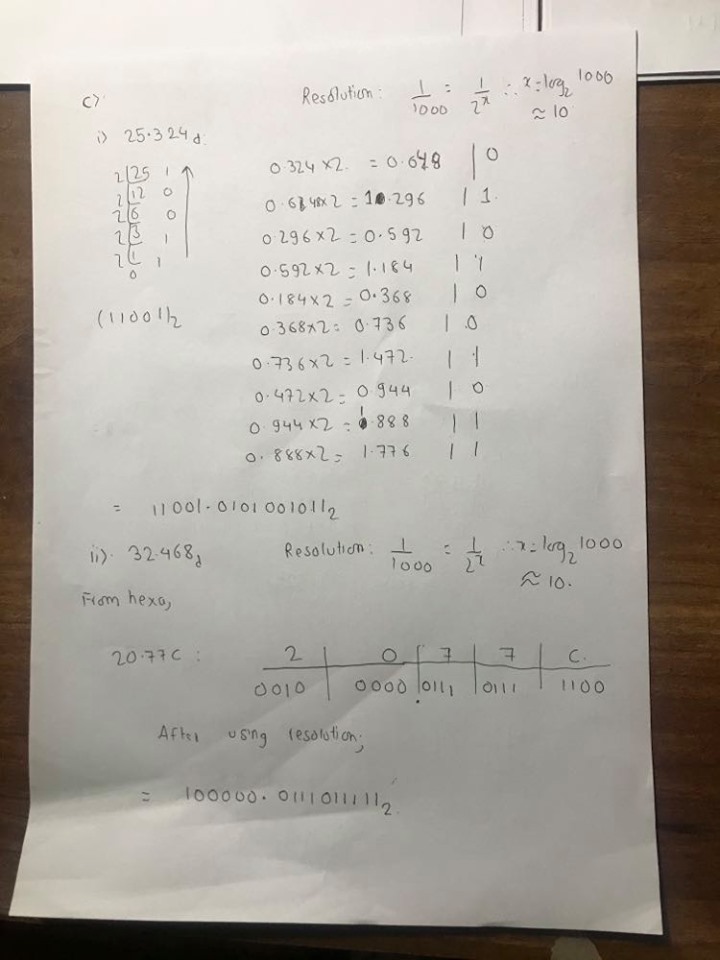
==================================================================================

**Problem 1 – (10 points) – Please be neat and show your work for BOTH numbers given below**

1. What is the resolution of the given **decimal** number?
2. Convert the number to hexadecimal with appropriate resolution. Show and explain the calculation needed to determine the number of hexadecimal places required in the converted hexadecimal number
3. Convert the number to binary with appropriate resolution. Show and explain the calculation needed to determine the number of binary places in the converted binary number

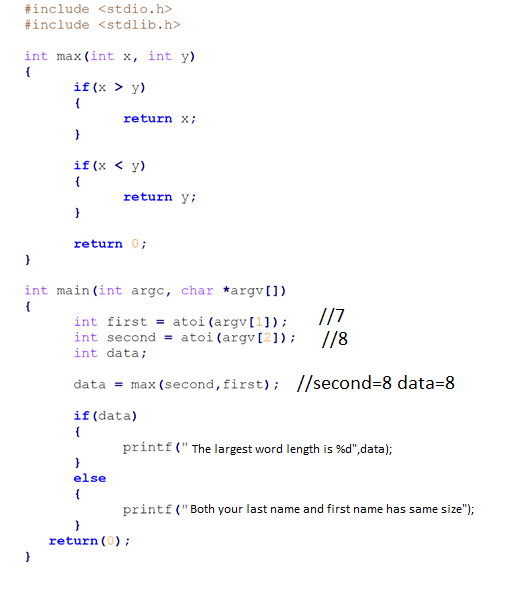
|  |  |
| --- | --- |
| 25.324d | 32.468d |

|\_ \_| represents the floor value.



**Problem 2 – (20 points)**

Analyze and carefully consider the purpose of the code given below. Complete the **two** incomplete printf statements in the main function. Each printf() statement should provide **appropriate information** to the user **if** it should execute. Assume that the code is built with gcc and produces the executable a.exe. When the code is executed in the terminal window, two parameters are provided. The first is the number of characters in your first name (decimal format). The second is the number of characters in your last name (decimal format). Use the terminal window to complete the execution input and generate the exact output from the program



|  |
| --- |
| $ ./a.exe 7 8  The largest word length is 8 |

**Problem 3** – (**15 points)**

Write the upper-case version of the first letter in your last name in the space here: H

Assume this character is then placed in the code below between the two single quotes and assigned to the variable maskp1.

char maskp7 = ‘H’ // 8 BIT VALUES

* What is the binary value of maskp7?

The binary value is: 01001000

//Char maskp7 is a 8 bit variable

* Assume that all the direction bits of the Port 1 direction register have been configured as output, describe what the following line of code will do.

P7OUT ^= maskp7;

Answer: This is for toggling the bits:

In MSP432, for port 7.1, if the output was pre-set to high, this will convert to low output. And if the output was pre-set to low mode, this will convert to high output on port 7.1

* What **specific device pins** on the MSP432P401R will be affected by the code above? (See problem 5 for the MSP432P401R schematic)

On the CPU of MSP432P401R, the device pin named: PM\_C0OUT/PM\_TA0CLK on the location 89.

**Problem 4** – (**20 points)** An initialization function is given below for an MSP432 system that uses four normally open single-pole/single-throw switches. The four switches are connected to four different MSP432 pins on one end of the switch. The other end of each switch is connected directly to either Vcc or GND.

* Fill in proper comments for every line of code.
* Use the schematic on the next page to properly connect the switches to each pin that is programmed in the initialization function. The switches must be connected in such a way that the software may use polling to detect the status of the switch.

**#define** BIT0 (0x0001)

**#define**  BIT1 (0x0002)

**#define** BIT2 (0x0004)

**#define** BIT3 (0x0008)

**#define**  BIT4 (0x0010)

**#define**  BIT5 (0x0020)

**#define** BIT6 (0x0040)

**#define** BIT7 (0x0080)

**void init(void)**

**{**

**P1DIR &= ~(BIT0 | BIT7); // P1.0 and P1.7 is set for input**

**P5DIR &= ~(BIT1 | BIT5); // P5.1 and P5.5 is set for input**

**P1REN |= BIT0 | BIT7; // Pull-Up/Pull Down configuration on P1.0 and P1.7 enabled**

**P5REN |= BIT1 | BIT5; // Pull-Up/Pull Down configuration on P5.1 and P5.5 enabled**

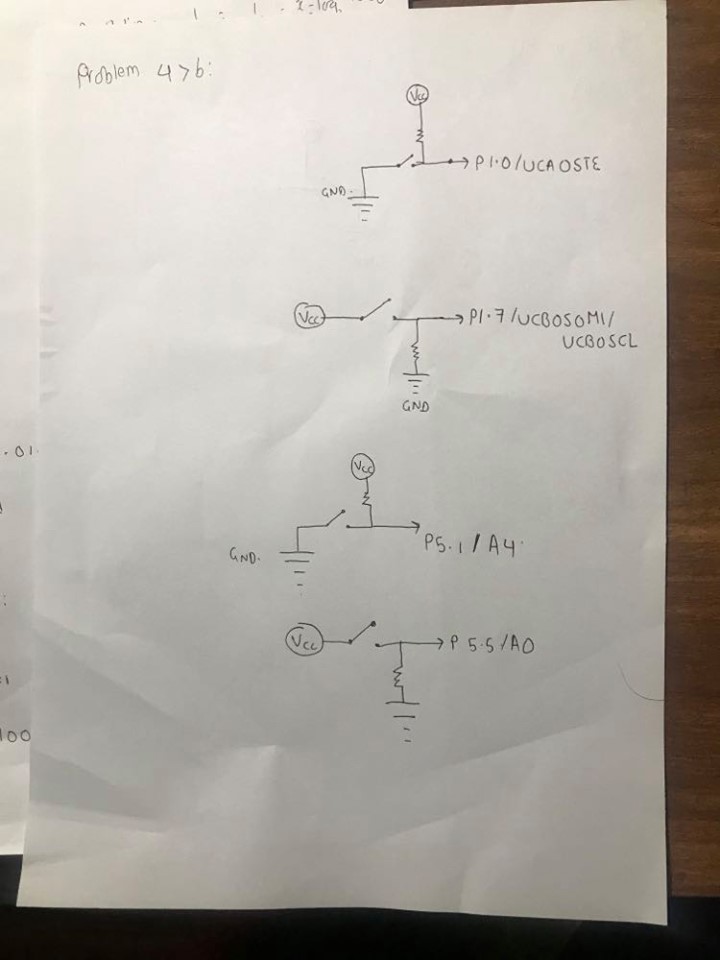
**P1OUT |= BIT0; // P1.0 set for Pull-up configuration**

**P1OUT &= ~BIT7; // P1.7 set for Pull-Down Configuration**

**P5OUT |= BIT1; // P5.1 is set for pull-up configuration**

**P5OUT &= ~BIT5; // P5.5 is set for pull-down configuration**

**}**



**In each cases, external resistor are mounted, and the Gnd represent the grounds. And VCC represent the voltage source. And the switch mounted is the external switches used.**



**Problem 5** – (**15 points)** Write **additional** code for the init() function in problem 5 **with comments** that would be needed if the program is going to use interrupts to detect events at each switch. Assume the board will boot up with no buttons pressed.

Void init(void){

//… all initialization for init routine has been made

P1->IES|=(BIT0); //port 1.0 set for interrupt from high to low transition

P1->IES&=~(BIT7); // port 1.7 is set for low to high transition

P1->IE|=(BIT0|BIT7); //enabled interrupt edge

P1->IFG=0; //cleared flag on interrupt for all ports

P5->IES|=(BIT1); //port 5.1 is set for interrupt from high to low transition

P5->IES&=~(BIT5);//port 5.5 is set for interrupt from low to high transition

P5->IE|=(BIT1|BIT5);// enabled interrupt edge

P5->IFG=0; // cleared flag on interrupt for all ports

NVIC->ISER[1]=1<<((PORT1\_IRQn)&31); // enable NVIC service routine on port 1

NVIC->ISER[1]=1<<((PORT5\_IRQn)&31); //enable NVIC service routine on port 5

PCM->CTL1 = PCM\_CTL0\_KEY\_VAL | PCM\_CTL1\_FORCE\_LPM\_ENTRY;

\_\_enable\_irq();//enable global interrupt

// Setting the sleep deep bit

SCB->SCR |= (SCB\_SCR\_SLEEPDEEP\_Msk);

// Do not wake up on exit from ISR

SCB->SCR |= SCB\_SCR\_SLEEPONEXIT\_Msk;

// Ensures SLEEPONEXIT takes effect immediately

\_\_DSB();

// Go to LPM4

\_\_sleep();

}

**Problem 6** – (**20 points)**

An MSP432 system that controls a counter is being developed and you may assume the following:

* A countDown() function exists that will decrement the counter display
* A countUp() function exists that will increment the counter display
* The counter is initialized to zero
* Two normally open SPST buttons (B1 and B2) exist that have been properly initialized to be read from GPIO pins on port 2
* A complete button event is defined as the press and release of the button.
* Both Digital IO pins measure a voltage of GND when the button is not pressed and a voltage of Vcc when the button is pressed
* Both Digital IO pins have been configured to cause an interrupt and are initially set to look for a rising edge event
* A global variable ‘mode’ exists that holds the current state of the pair of buttons regarding what edge type will trigger an interrupt from each button. The states are documented below.

|  |
| --- |
| **B1 - P2.2 (increment)**  **B2 - P2.5 (decrement)** |

**You must add code to the ISR on the next page:**

* **To properly configure the interrupt and flag resisters.**
* **Keep track of and update the mode and counter display**
* **The counter display must be updated on the button press (not the release)**
* **Add comments to indicate where there is a button ‘press’ and a ‘release’**

**BONUS (5 points) – Comment on what happens when both buttons are pressed at exactly the same time**

// Global variable holding the current button edge detection state

char mode **=** 0**;**

// Keep track of the state of the buttons

// These states have meaning when the CPU is asleep

// 0 - Both buttons respond to rising edge

// 1 - B1 responds to rising edge/B2 responds to falling edge

// 2 - B1 responds to falling edge/B2 responds to rising edge

// 3 - Both buttons respond to falling edge

**/\* Port 2 interrupt to service button events \*/**

**void PORT2\_IRQHandler(void){**

**P2IE &=~(BIT2|BIT5) //disabling the interrupt for port 5.2 and 5.5**

**\_\_delay\_cycles(96000); //avoiding bouncing**

**if(P2IFG & BIT2){**

**P2IES^=BIT2;** //toggling the P2.2, so if button is pressed, it’s saying the //button to get ready to counter increment

**if(mode == 0 ){**

**mode = 3; // we do not respond to rising event**

**}**

**else if(mode == 2 ){**

**mode = 1; //we responded to falling event on switch B1**

**countUp(); // counting up**

**}**

**else if(mode == 3 ){**

**mode = 0; //we responded to falling event on both switch**

**countUp();**

**}**

**else{**

**mode = 2; //we do not respond to rising event**

**}**

**}**

**else{**

**P2IES^=BIT5;** //toggling the P2.5, so if button is pressed, it’s saying the //button to get ready to counter decrement

**if(mode == 0 ){**

**mode = 3; // we do not respond to rising event**

**}**

**else if(mode ==2 ){**

**mode = 1; // we do not respond to rising event for B2**

**}**

**else if(mode == 3 ){**

**mode = 0;**

**countDown(); // we responded to falling event by counter //decrementing**

**}**

**else{**

**mode = 2;**

**countDown(); // we responded to falling event by counter //decrementing**

**}**

**}**

**P2IFG&=~(BIT2|BIT5); //clearing all the flags**

**P2IE|=(BIT2|BIT5); //enabling interrupt service**

**}**